Remoting an old APC PDU using SNMP and remOcular

Garage sale at the office: Good chance to grab on some (very) old hardware, like an APC 9221 PDU. Yes, it's old (some 15 years or so), so surely not state of the art. But yet good enough to use in my home lab. Who could tell that there were some unforeseen issues waiting for me ...

Well, as always is with old hardware, while they still may work, there's a good chance to run into issues.
With these old APC 92xx series PDU, there's many: Not only do they lack an API of any sort, the remoting via Telnet is just like stoneage. And even worse: The web gui, which is Java-driven, just fails to run due to some very obscure JRE runtime errors. Bummer!

However, I felt a need to have a nice web frontend. Then I remembered the not so commonly known remOcular (sic!) by Tobi Oetiker.
So I did some hackery to write a plugin to control this old dinosaur from within remOcular.
It's far from perfect, but does the job well enough for me, allowing me to easily power my lab devices through the web browser.

To use it, download the ApcPdu plugin into remOcular's **/path/to/remOcular/lib/Plugin/** folder, e.g.:
wget http://phaq.phunsites.net/files/2016/04/ApcPdu.pm_.gz -O- | gunzip > /path/to/remOcular/lib/Plugin/ApcPdu.pm

Then register it with the config file like so:
--snip--
*** Plugin ***

+remOcular::Plugin::ApcPdu
--snip--

I spared the hassle of doing the config file handling stuff, add settings directly in **ApcPdu.pm** by editing the config section. These should be self-explanatory.

```
# ==== START OF CONFIG SECTION ============================================================ #

# configure PDUs, register them as follows
#
# "<FQDN or IP>" => { "snmpv1_community" => "<COMMUNITY>", "outlets" => <0 = AUTO, or give exact nummber>,
"enabled" => <0 or 1> },
#
my %pducfg = (
        "somehost.local" => { "snmpv1_community" => "public", "outlets" => 0, "enabled" => 1 },
);



# configure SNMP OIDs
#
my %oids = (
        "outletDesc"    => 'enterprises.318.1.1.4.5.2.1.3',
        "outletState"   => 'enterprises.318.1.1.4.4.2.1.3',
);



# power states
```

```
#
my @pwrstate = ( 'undefined', 'ON', 'OFF' );


# path to snmp utilities
#
my $bin_snmpget = '/usr/local/bin/snmpget';
my $bin_snmpset = '/usr/local/bin/snmpset';


# open distinct errorlog for the plugin
#
my $errorlog = Mojo::Log->new (
    path  => '/var/log/remocular/ApcPdu.log',
    level => 'debug',
);
```

Note that I'm assuming **/var/log/remocular/** to exist, so a debug logfile will be created there at runtime.
The logging is rather chatty, and looks somewhat like this:
Wed Apr 20 09:19:17 2016] [debug] PDU somehost.local requests outlet auto-detection
[Wed Apr 20 09:19:17 2016] [debug] somehost.local: detected 8 outlets
----snip----
[Tue Apr 26 06:31:46 2016] [debug] somehost.local: setting outlet 8 to state ON
----snip---
[Tue Apr 26 06:34:26 2016] [debug] somehost.local: getting outlet 6 state ...
[Tue Apr 26 06:34:26 2016] [debug] raw output: SNMPv2-SMI::enterprises.318.1.1.4.5.2.1.3.6 = STRING: "Cisco 3750G"
[Tue Apr 26 06:34:26 2016] [debug] raw output: SNMPv2-SMI::enterprises.318.1.1.4.4.2.1.3.6 = INTEGER: 2

The plugin doesn't offer much functionality. Besides trying to auto-detect outlet numbers during initialization, it offers basic functionality to retrieve current outlet state and performing power ON/OFF operations.

All operations can be either performed on ALL or a single outlet.

And here some screenies on what the frontend looks like: